How to Use Dropout Correctly on Residual Networks with Batch Normalization

Bum Jun Kim[1] **Hyeyeon Choi**[1] **Hyeonah Jang**[1] **Donggeon Lee**[1] **Sang Woo Kim**[1]

[1]Department of Electrical Engineering, Pohang University of Science and Technology, Pohang, South Korea

Abstract

For the stable optimization of deep neural networks, regularization methods such as dropout and batch normalization have been used in various tasks. Nevertheless, the correct position to apply dropout has rarely been discussed, and different positions have been employed depending on the practitioners. In this study, we investigate the correct position to apply dropout. We demonstrate that for a residual network with batch normalization, applying dropout at certain positions increases the performance, whereas applying dropout at other positions decreases the performance. Based on theoretical analysis, we provide the following guideline for the correct position to apply dropout: apply one dropout after the last batch normalization but before the last weight layer in the residual branch. We provide detailed theoretical explanations to support this claim and demonstrate them through module tests. In addition, we investigate the correct position of dropout in the head that produces the final prediction. Although the current consensus is to apply dropout after global average pooling, we prove that applying dropout before global average pooling leads to a more stable output. The proposed guidelines are validated through experiments using different datasets and models.

1 INTRODUCTION

Deep neural networks have demonstrated remarkable performance across a range of fields including computer vision and natural language processing. Previously, training a deep neural network using a large number of parameters was known to be difficult owing to the overfitting problem. To address this issue, several regularizers such as dropout, batch normalization (BN), and label smoothing have recently been proposed [Ioffe and Szegedy, 2015, Srivastava et al., 2014, Szegedy et al., 2016]. They have made significant contributions to the stable optimization of deep neural networks and have been widely used in various tasks.

For the architectural design of modern neural networks, the [BN–ReLU–Weight] pipeline has been widely used, where dropout can be added. However, there remains a lack of consensus regarding the correct position for applying dropout, and practitioners have chosen different positions. For example, Pham and Le [2021], Isola et al. [2017], Romera et al. [2018], Yan et al. [2018] applied dropout **after every BN**; however, in the studies of Cai et al. [2019], Qi et al. [2017], Pavllo et al. [2019], Li et al. [2019], Zagoruyko and Komodakis [2016], Zhan et al. [2020], dropout was applied **after each ReLU**. Furthermore, Castro et al. [2021], Ghiasi et al. [2018] used dropout **after the weight layers** and Ravi and Larochelle [2017], Lim et al. [2016], Liu et al. [2020] applied dropout **after every MaxPool layer**. Based on these practices, we highlight the need for further research to determine the correct position to apply dropout.

In fact, He et al. [2016b] empirically found that applying dropout at the output of the residual block decreased the performance, whereas Zagoruyko and Komodakis [2016] reported that applying dropout inside the residual branch improved the performance. These observations highlight the importance of selecting the correct position for dropout. That is, choosing the incorrect order of layers implies a potential performance decrease, and if dropout is placed in the correct position, a potential performance improvement can be obtained at little extra cost.

Moreover, there is a lack of theoretical analysis to determine the correct position to apply dropout. As an exception, only the study by Li et al. [2019] theoretically discussed the position of applying dropout. Their analysis advocated using dropout before each weight layer to harmonize the dropout and BN. However, we present the limitations and a reinterpretation of their study. For example, starting from their analysis and considering residual networks, we derive

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

a different conclusion regarding the correct position for the dropout.

In this study, we investigate the correct position of dropout. First, we analyze the different dropout operations in the training and test phases, which are harmful to the normalization step of BN. We quantify the different behaviors of dropout in the training and test phases as an inconsistency ratio and argue that the inconsistency ratio is influenced by the order of the layers. Considering this phenomenon, we discuss the best order of layers to mitigate the inconsistency ratio. Our conclusions suggest that dropout and ReLU are permutable (Proposition 1); using dropout before the weight layer resolves the inconsistency ratio under certain weight conditions (Proposition 2); and residual blocks mitigate the inconsistency ratio better than non-residual blocks for certain positions (Propositions 3 and 4). Based on these analyses, we propose applying one dropout after the last BN but before the last weight layer in the residual branch (Guideline 1).

In addition, we analyze the use of dropout in the head that outputs the final prediction. Although the current consensus is to apply dropout after global average pooling (GAP), we prove that using dropout before GAP leads to a more stable output (Proposition 5). Based on this analysis, we propose the use of dropout before GAP (Guideline 2).

The validities of Guidelines 1 and 2 are verified through experiments on different datasets including CIFAR-$\{10, 100\}$, Caltech-101, Oxford IIIT-Pet, and ImageNet. We observed that the performance of the model improved with dropout when the guidelines were followed.

2 THEORETICAL ANALYSIS

Notation In this paper, we use the notations $\mathrm{E}[x_i]$ and $\mathrm{Var}[x_i]$ to denote the mean and variance of the ith element x_i of vector $\mathbf{x}$ over the mini-batch. To represent $\mathrm{Var}[x_i]$ for an arbitrary index i, we use the abbreviation $\mathrm{Var}[\mathbf{x}]$.

2.1 BACKGROUND

Dropout is an operation that randomly drops certain features in the target layer during training [Srivastava et al., 2014]. First, we define the Dropout operation as follows.

Definition 1. *Operation Dropout with a keep probability $p \in (0,\ 1)$ is defined as follows:*

$$\mathrm{Dropout}_{\mathrm{train}}(\mathbf{x}) \coloneqq \frac{1}{p}\mathbf{M}\mathbf{x}, \tag{1}$$

$$\mathrm{Dropout}_{\mathrm{test}}(\mathbf{x}) \coloneqq \mathbf{x}, \tag{2}$$

where $\mathbf{x}$ is an n-dimensional vector and $\mathbf{M}$ is an $n \times n$ diagonal matrix with $m_{ij} = 0$ for $i \neq j$ and $m_{ij} \sim \mathrm{Bernoulli}(p)$ for $i = j$.

According to the Bernoulli distribution, $m_{i,i}$ is either one with keep probability p or zero with drop probability $1 - p$, and is independent of $\mathbf{x}$. Thus, we have $\mathrm{E}[m_{i,i}^2] = \mathrm{E}[m_{i,i}] = p$ and $\mathrm{E}[m_{i,i}m_{j,j}] = p^2$ for $i \neq j$. This property ensures mean consistency in the training and test phases, *i.e.*, $\mathrm{E}[\frac{1}{p}\mathbf{M}\mathbf{x}] = \mathrm{E}[\mathbf{x}]$.

However, dropout does not provide variance consistency in the training and test phases. To investigate this phenomenon, we introduce the following inconsistency ratio:

Definition 2. *Let $f(\mathbf{x})$ be the output feature of an operation f. The inconsistency ratio $\Delta(f(\mathbf{x}))$ is defined as the ratio of the variance of $f(\mathbf{x})$ between the training and test phases.*

$$\Delta(f(\mathbf{x})) \coloneqq \frac{\mathrm{Var}[f_{\mathrm{test}}(\mathbf{x})]}{\mathrm{Var}[f_{\mathrm{train}}(\mathbf{x})]}. \tag{3}$$

For example, $\Delta(f(\mathbf{x})) = 0.5$ indicates that the variance of $f(\mathbf{x})$ during the training phase is twice as large as that during the test phase. To obtain variance consistency, we should achieve $\Delta(f(\mathbf{x})) = 1$.

Li et al. [2019] state that the use of dropout yields variance inconsistency in a neural network. During the training phase,

$$\mathrm{Var}[\mathrm{Dropout}_{\mathrm{train}}(\mathbf{x})] \tag{4}$$

$$= \mathrm{E}[\frac{1}{p^2}m_{i,i}^2 x_i^2] - (\mathrm{E}[\frac{1}{p}m_{i,i}x_i])^2 \tag{5}$$

$$= \frac{1}{p}\,\mathrm{Var}[x_i] + \frac{1-p}{p}(\mathrm{E}[x_i])^2, \tag{6}$$

which is greater than $\mathrm{Var}[\mathrm{Dropout}_{\mathrm{test}}(\mathbf{x})] = \mathrm{Var}[x_i]$ for $p < 1$. Thus, $\Delta(\mathrm{Dropout}(\mathbf{x})) < 1$.

The variance inconsistency of dropout causes a problem when we use dropout with BN. Although subsequent BN anticipates receiving the same mean and variance during the training and test phases, the variance inconsistency of dropout provides different variances to BN during the training and test phases. For example, consider an input feature $\mathbf{h}$ to BN where $\mathrm{Var}[\mathbf{h}_{\mathrm{train}}] = 10$, $\mathrm{Var}[\mathbf{h}_{\mathrm{test}}] = 2$, $\mathrm{E}[\mathbf{h}_{\mathrm{train}}] = 0$, and $\mathrm{E}[\mathbf{h}_{\mathrm{test}}] = 0$. The normalization step of BN uses the mean and variance of the training phase to produce $\frac{\mathbf{h}}{\sqrt{10}}$, which is also used in the test phase because BN assumes the same mean and variance. After the normalization step, we obtain $\mathrm{Var}[\frac{\mathbf{h}_{\mathrm{train}}}{\sqrt{10}}] = 1$ during the training phase. However, during the test phase, the BN receives a feature with a different variance, resulting in $\mathrm{Var}[\frac{\mathbf{h}_{\mathrm{test}}}{\sqrt{10}}] = 0.2$. Thus, the variance inconsistency breaks the consistent behavior of the subsequent BN during the training and test phases. This phenomenon explains the decrease in performance when dropout and BN are used simultaneously.

2.2 ORDER OF OPERATIONS

A modern neural network is composed of numerous operations such as ReLU, weight layer, BN, and skip connection,

whose output feature map is a potential position for applying dropout. Here, we claim that the position of applying dropout influences the inconsistency ratio. The goal of this study is to investigate the best position for applying dropout that offers a $\Delta(f(\mathbf{x}))$ close to one, which harmonizes the dropout with BN. First, we discuss the order of the operations.

Order of Dropout and ReLU Some practitioners have applied dropout before ReLU, whereas others have applied dropout after ReLU (Section 1). Here, we claim that the influence of the order of ReLU and dropout is insignificant.

Proposition 1. *ReLU and dropout operations are permutable:*

$$\mathrm{ReLU}(\mathrm{Dropout}(\mathbf{x})) = \mathrm{Dropout}(\mathrm{ReLU}(\mathbf{x})). \quad (7)$$

Proof. First, during the test phase, dropout operates as an identity function that satisfies Eq. 7. Secondly, we claim that, even during the training phase, the influence of the order of ReLU and dropout is insignificant. Consider that we sampled matrix $\mathbf{M}$ to denote $\mathrm{Dropout}^{\mathbf{M}}_{\mathrm{train}}(\mathbf{x})$. Because matrix $\mathbf{M}$ is a diagonal matrix, the ith element of vector $\mathrm{ReLU}(\mathrm{Dropout}^{\mathbf{M}}_{\mathrm{train}}(\mathbf{x}))$ can be written as

$$[\mathrm{ReLU}(\mathrm{Dropout}^{\mathbf{M}}_{\mathrm{train}}(\mathbf{x}))]_i = \mathrm{ReLU}\Big(\frac{1}{p} m_{i,i} x_i\Big). \quad (8)$$

Here, the coefficient $m_{i,i}/p$ is a non-negative scalar. For $\mathrm{ReLU}(x) = \max(0,\ x)$, we know that $\mathrm{ReLU}(kx) = k\,\mathrm{ReLU}(x)$ for a non-negative scalar k. Thus, we obtain

$$\mathrm{ReLU}\Big(\frac{1}{p} m_{i,i} x_i\Big) = \frac{1}{p} m_{i,i}\,\mathrm{ReLU}(x_i) \quad (9)$$

$$= [\mathrm{Dropout}^{\mathbf{M}}_{\mathrm{train}}(\mathrm{ReLU}(\mathbf{x}))]_i. \quad (10)$$

Therefore, we conclude that $\mathrm{ReLU}(\mathrm{Dropout}^{\mathbf{M}}_{\mathrm{train}}(\mathbf{x})) = \mathrm{Dropout}^{\mathbf{M}}_{\mathrm{train}}(\mathrm{ReLU}(\mathbf{x}))$. □

In summary, the order in which dropout and ReLU operations are applied to vector $\mathbf{x}$ does not influence the result. In the remainder of this paper, we do not consider applying dropout before ReLU unless specified otherwise.

However, commutativity with dropout does not hold for other operations such as weight layer and BN. We further investigate the effects of the order of these operations.

Order of Dropout and Weight We refer to dropout before the weight layer as *PreDropout*. For PreDropout, we have $\mathbf{W}\,\mathrm{Dropout}_{\mathrm{train}}(\mathbf{x}) = \frac{1}{p}\mathbf{WMx}$ and can interpret the two operations using another weight $\mathbf{WM}/p$. Similarly, for the PostDropout order, we write $\mathrm{Dropout}_{\mathrm{train}}(\mathbf{Wx}) = \frac{1}{p}\mathbf{MWx}$. The difference between PreDropout and PostDropout occurs because $\mathbf{WM} \neq \mathbf{MW}$ for $p < 1$. These matrices can be represented as

$$\mathbf{WM} = \begin{bmatrix} | & | & | \\ m_{1,1}w_1 & \cdots & m_{n,n}w_n \\ | & | & | \end{bmatrix},$$

$$\mathbf{MW} = \begin{bmatrix} — & m_{1,1}w_1 & — \\ — & \cdots & — \\ — & m_{m,m}w_m & — \end{bmatrix}.$$

The diagonal element $m_{i,i}$ is either zero or one. Thus, PreDropout is equivalent to dropping *columns* in the weight matrix $\mathbf{W}$ with $1/p$ constant scaling, whereas PostDropout is equivalent to dropping *rows* in the weight matrix $\mathbf{W}$ with $1/p$ constant scaling. Thus, the characteristics of PreDropout and PostDropout differ for $p < 1$.

The question then arises as to which is more effective in reducing variance inconsistency. Li et al. [2019] suggest that for the PreDropout order, increasing the width alleviates variance inconsistency, assuming a certain condition on weight. However, we find that increasing the width does not solve the variance inconsistency for other weight conditions such as He initialization [He et al., 2015]. Although Li et al. [2019] emphasized increasing the width, we focus more on the weight condition. Our reinterpretation of their study is as follows.

Proposition 2. *PreDropout exhibits less variance inconsistency than PostDropout*

$$\Delta(\underbrace{\mathrm{Dropout}(\mathbf{Wx})}_{\textit{PostDropout}}) < \Delta(\underbrace{\mathbf{W}\,\mathrm{Dropout}(\mathbf{x})}_{\textit{PreDropout}}) < 1, \quad (11)$$

where the first inequality holds if and only if $\sum_{j=1}^{n}\sum_{k\neq j}^{n} w_{i,j}w_{i,k}\,\mathrm{E}[x_j x_k] > 0$.

According to Proposition 2, the advantage of PreDropout depends on the weight condition. For example, if the weight has a nonzero mean and $\mathbf{x}$ comes from ReLU output, the condition holds and PreDropout is advantageous. However, for zero-mean weight, the inconsistency ratios of PostDropout and PreDropout can be indistinguishable. A detailed proof can be found in the Appendix.

Empirical Observation We measured the two inconsistency ratios, $\Delta(\mathrm{Dropout}(\mathbf{Wx}))$ and $\Delta(\mathbf{W}\,\mathrm{Dropout}(\mathbf{x}))$. The [BN–ReLU–Weight–BN–ReLU] pipeline, which is commonly deployed in the residual branch, produced output $\mathbf{x}$. The input to the pipeline was sampled from $\mathcal{N}(0,\ 1)$ with a mini-batch size of 10^5. We tested five cases of width n from $\{128,\ 256,\ 512,\ 1024,\ 2048\}$. We used a keep probability of 0.5 for dropout. For the weight condition, we set $\mathrm{Var}[\mathbf{W}] = 2/n$, similar to the He initialization but varied $\mathrm{E}[\mathbf{W}]$.

The results are summarized in Figure 1. We observed that $\Delta(\mathrm{Dropout}(\mathbf{Wx})) < \Delta(\mathbf{W}\,\mathrm{Dropout}(\mathbf{x})) < 1$ for

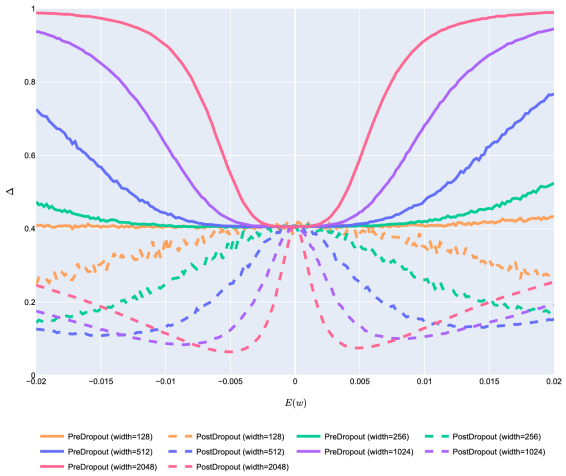

Figure 1: Empirical validation on Proposition 2. We observed $\Delta(\mathrm{Dropout}(\mathbf{Wx})) < \Delta(\mathbf{W}\,\mathrm{Dropout}(\mathbf{x})) < 1$ for $\mathrm{E}[\mathbf{W}] \neq 0$.

$\mathrm{E}[\mathbf{W}] \neq 0$. The advantage of using PreDropout is visible in both $\mathrm{E}[\mathbf{W}] > 0$ and $\mathrm{E}[\mathbf{W}] < 0$. This is because $\mathrm{E}[x_j x_k] \geq 0$ for $\mathbf{x}$ from the ReLU output and $\sum_{j=1}^{n}\sum_{k\neq j}^{n} w_{i,j}w_{i,k} \approx n(n-1)(\mathrm{E}[\mathbf{W}])^2 > 0$ for large $|\mathrm{E}[\mathbf{W}]|$. A large width has little effect if $\mathrm{E}[\mathbf{W}] = 0$. Thus, we found that the advantage of PreDropout requires $\mathrm{E}[\mathbf{W}] \neq 0$ and intensifies as the width increases.

Therefore, the advantage of the PreDropout is dependent on the weight condition. For example, if the weight is polarized to a large $|\mathrm{E}[\mathbf{W}]|$ through training, the accumulation of its products becomes positive, allowing us to enjoy the advantage of PreDropout. Li et al. [2019] empirically observed that the trained weight satisfies a certain condition to advocate PreDropout. We conjecture that the condition holds and validate the superior performance of PreDropout over PostDropout through experiments (Section 3).

However, we later demonstrate cases where neither PreDropout nor PostDropout improves performance. Rather than comparing PreDropout and PostDropout, we find that the properties of residual networks have a greater influence on the alleviation of variance inconsistency.

2.3 DROPOUT IN RESIDUAL BLOCK

A residual network is composed of residual blocks, which consist of a residual and skip branch. PreResNet, also known as ResNetV2, is a variant that applies BN first in the residual branch [He et al., 2016b]. In this section, we provide an analysis of PreResNet, which can be extended to other variants of residual networks such as ResNetV1 [He et al., 2016a]. We examine eight possible positions to apply dropout, labeled P0–P7 (Figure 2).

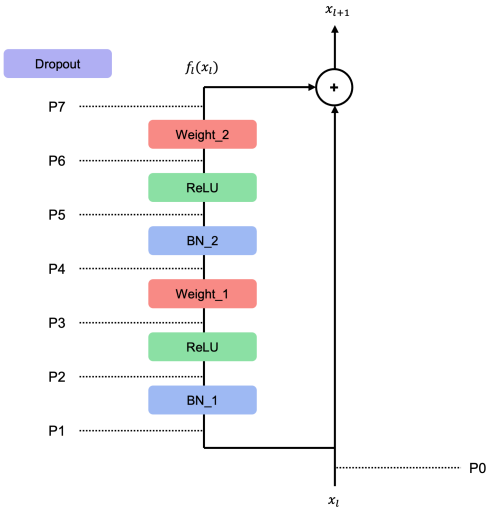

Figure 2: Residual block of PreResNet.

First, we consider applying dropout at P1. As mentioned earlier, the use of dropout causes inconsistency in the input variance of the next BN. If we apply dropout at P1, it directly influences the input variance of the first BN of the lth residual block. Similarly, applying dropout at one of (P2, P3, or P4) results in an inconsistency in the input variance of the second BN of the lth residual block. Furthermore, applying dropout at P0 causes variance inconsistency in the first BN of the lth residual block.

However, applying dropout at one of (P5, P6, or P7) has distinct characteristics. The use of dropout at one of (P5, P6, or P7) causes inconsistency in the input variance of the BN at the next residual block, *i.e.*, the first BN of the $(l+1)$th residual block. Because the output of the residual branch is merged with the skip connection, the variance inconsistency in the next residual block behaves differently compared to the other positions. From this observation, we investigate the inconsistency ratio of dropout at (P5, P6, or P7) in detail.

First, we analyze the output $f_l(\mathbf{x}_l)$ of the [BN–ReLU–Weight–BN–ReLU–Dropout–Weight] pipeline from the input feature map $\mathbf{x}_l$ of lth block, considering dropout at P6. Note that BN outputs $\gamma\hat{x} + \beta$ from the normalized feature $\hat{x}$. For the lth residual block, let the output of the second BN be $\mathbf{z}_l \sim \mathcal{N}(0,\ \gamma_l^2)$.[1] This feature map passes through ReLU, weight, and skip connection. First, we know

$$\mathrm{E}[\mathrm{ReLU}(\mathbf{z}_l)] = \frac{1}{\sqrt{2\pi}}\gamma_l, \tag{12}$$

$$\mathrm{Var}[\mathrm{ReLU}(\mathbf{z}_l)] = \frac{\pi-1}{2\pi}\gamma_l^2. \tag{13}$$

[1] At initialization, BN has $\gamma = 1$ and $\beta = 0$. During training, although γ becomes a specific value, β stays close to zero. We conjecture that β near zero is advantageous to preserving zero-centered ReLU input. From this observation, we allow a degree of freedom in γ; however, we use $\beta = 0$.

See the Appendix for details of the above equations. Now, we apply dropout. During the training phase,

$$\mathrm{E}[\mathrm{Dropout}_{\mathrm{train}}(\mathrm{ReLU}(\mathbf{z}_l))] = \frac{1}{\sqrt{2\pi}}\gamma_l, \tag{14}$$

$$\mathrm{Var}[\mathrm{Dropout}_{\mathrm{train}}(\mathrm{ReLU}(\mathbf{z}_l))] = \frac{\pi/p-1}{2\pi}\gamma_l^2. \tag{15}$$

Finally, we apply weight that is initialized from He initialization $\mathcal{N}(0,\ 2/n)$. Then, we have

$$\mathrm{E}[\mathbf{W}\,\mathrm{Dropout}_{\mathrm{train}}(\mathrm{ReLU}(\mathbf{z}_l))] = 0, \tag{16}$$

$$\mathrm{Var}[\mathbf{W}\,\mathrm{Dropout}_{\mathrm{train}}(\mathrm{ReLU}(\mathbf{z}_l))] = \frac{\pi/p-1}{\pi}\gamma_l^2, \tag{17}$$

which represents the variance of the residual branch $f_l(\mathbf{x}_l)$:

$$\mathrm{Var}[f_{l,\mathrm{train}}(\mathbf{x}_l)] = \frac{\pi/p-1}{\pi}\gamma_l^2, \tag{18}$$

$$\mathrm{Var}[f_{l,\mathrm{test}}(\mathbf{x}_l)] = \frac{\pi-1}{\pi}\gamma_l^2. \tag{19}$$

The same result can be obtained when applying dropout at P5 (Proposition 1) and P7 (Proposition 2 for the zero-mean weight). Now, consider two choices for the building blocks: non-residual block $f_l(\mathbf{x}_l)$ and residual block $\mathbf{x}_l + f_l(\mathbf{x}_l)$. We begin by investigating the case where $l = 0$.

Proposition 3. *For the* 0*th very first block, choosing a residual block alleviates variance inconsistency from dropout when compared with a non-residual block*

$$\Delta(\underbrace{f_0(\mathbf{x}_0)}_{\textit{Non-residual}}) < \Delta(\underbrace{\mathbf{x}_0 + f_0(\mathbf{x}_0)}_{\textit{Residual}}) < 1, \tag{20}$$

if we apply dropout at one of (P5, P6, or P7) in PreResNet.

Proof. Note that for $x > 0$, $y > 0$, and $c > 0$, if $\frac{x}{y} < 1$, then $\frac{x}{y} < \frac{x+c}{y+c}$. Using this inequality, we obtain

$$\Delta(f_0(\mathbf{x}_0)) = \frac{\mathrm{Var}[f_{0,\mathrm{test}}(\mathbf{x}_0)]}{\mathrm{Var}[f_{0,\mathrm{train}}(\mathbf{x}_0)]} \tag{21}$$

$$< \frac{\mathrm{Var}[\mathbf{x}_0] + \mathrm{Var}[f_{0,\mathrm{test}}(\mathbf{x}_0)]}{\mathrm{Var}[\mathbf{x}_0] + \mathrm{Var}[f_{0,\mathrm{train}}(\mathbf{x}_0)]} \tag{22}$$

$$= \Delta(\mathbf{x}_0 + f_0(\mathbf{x}_0)) < 1. \tag{23}$$

The advantage of choosing a residual block appears when the skip connection is located after dropout but before the subsequent BN. Thus, applying dropout at one of (P5, P6, or P7) alleviates variance inconsistency. Others, such as (P2, P3, or P4) correspond to non-residual blocks and do not alleviate variance inconsistency. □

The above derivation exploits the fact that, for the very first block, the input feature map $\mathbf{x}_0$ exhibits no variance inconsistency. However, when we choose a residual block, subsequent blocks receive the input feature map $\mathbf{x}_l$, which exhibits variance inconsistency due to dropout. Nonetheless, even in this scenario, choosing a residual block is still advantageous for reducing variance inconsistency.

Proposition 4. *For the* l*th block, if all* l'*th blocks for* $l' < l$ *are residual blocks, then choosing a residual block alleviates variance inconsistency from dropout when compared to a non-residual block*

$$\Delta(\underbrace{f_l(\mathbf{x}_l)}_{\textit{Non-residual}}) < \Delta(\underbrace{\mathbf{x}_l + f_l(\mathbf{x}_l)}_{\textit{Residual}}) < 1, \tag{24}$$

if we apply dropout at one of (P5, P6, or P7) in PreResNet.

Proof. The skip connection adds the result of the residual branch as $\mathbf{x}_{l+1} = \mathbf{x}_l + f_l(\mathbf{x}_l)$. As De and Smith [2020], Brock et al. [2021] describe, the residual block accumulates its variance:

$$\mathrm{Var}[\mathbf{x}_{l+1}] = \mathrm{Var}[\mathbf{x}_l] + \mathrm{Var}[f_l(\mathbf{x}_l)]. \tag{25}$$

Thus, $\mathbf{x}_l$ is the accumulation of the residual branches from 0 to $l-1$. For the training and test phases,

$$\mathrm{Var}[\mathbf{x}_{l,\mathrm{train}}] = \mathrm{Var}[\mathbf{x}_0] + \frac{\pi/p-1}{\pi}\sum_{i=0}^{l-1}\gamma_i^2, \tag{26}$$

$$\mathrm{Var}[\mathbf{x}_{l,\mathrm{test}}] = \mathrm{Var}[\mathbf{x}_0] + \frac{\pi-1}{\pi}\sum_{i=0}^{l-1}\gamma_i^2. \tag{27}$$

First, if we choose a non-residual block for the lth block, we obtain $f_l(\mathbf{x}_l)$ and its inconsistency ratio as

$$\Delta(f_l(\mathbf{x}_l)) = \frac{\mathrm{Var}[f_{l,\mathrm{test}}(\mathbf{x}_l)]}{\mathrm{Var}[f_{l,\mathrm{train}}(\mathbf{x}_l)]} \tag{28}$$

$$= \frac{\frac{\pi-1}{\pi}\gamma_l^2}{\frac{\pi/p-1}{\pi}\gamma_l^2} = \frac{\pi-1}{\pi/p-1}. \tag{29}$$

Second, if we choose a residual block for the lth block, we obtain $\mathbf{x}_l + f_l(\mathbf{x}_l)$ and its inconsistency ratio as

$$\Delta(\mathbf{x}_l + f_l(\mathbf{x}_l)) = \Delta(\mathbf{x}_{l+1}) = \frac{\mathrm{Var}[\mathbf{x}_{l+1,\mathrm{test}}]}{\mathrm{Var}[\mathbf{x}_{l+1,\mathrm{train}}]} \tag{30}$$

$$= \frac{\mathrm{Var}[\mathbf{x}_0] + \frac{\pi-1}{\pi}\sum_{i=0}^{l}\gamma_i^2}{\mathrm{Var}[\mathbf{x}_0] + \frac{\pi/p-1}{\pi}\sum_{i=0}^{l}\gamma_i^2} < 1. \tag{31}$$

Finally, it is known that for $x > 0$, $y > 0$, and $c > 0$, if $\frac{x}{y} < 1$, then $\frac{x}{y} < \frac{x+c}{y+c}$. Using this inequality, we obtain

$$\Delta(\mathbf{x}_l + f_l(\mathbf{x}_l)) > \frac{\frac{\pi-1}{\pi}\sum_{i=0}^{l}\gamma_i^2}{\frac{\pi/p-1}{\pi}\sum_{i=0}^{l}\gamma_i^2} = \frac{\pi-1}{\pi/p-1} \tag{32}$$

$$= \Delta(f_l(\mathbf{x}_l)), \tag{33}$$

which concludes the proof of this proposition. □

The difference between the two inconsistency ratios is due to $\mathrm{Var}[\mathbf{x}_0]$. Next, we empirically test the effect of $\mathrm{Var}[\mathbf{x}_0]$.

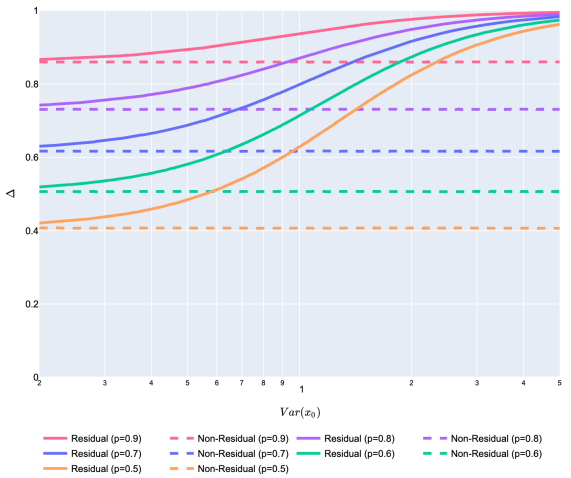

Figure 3: Empirical validation on Propositions 3 and 4.

Empirical Observation We measured the two inconsistency ratios, $\Delta(f_l(\mathbf{x}_l))$ and $\Delta(\mathbf{x}_l + f_l(\mathbf{x}_l))$. We tested five different keep probabilities $\{0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9\}$ for dropout. We applied dropout at P6 to construct the [BN–ReLU–Weight–BN–ReLU–Dropout–Weight] pipeline. We used a mini-batch size of 10^5 with a width n of 128. We initialized the weights using the He initialization $\mathcal{N}(0,\ 2/n)$. We varied $\mathrm{Var}[\mathbf{x}_0]$ and observed two inconsistency ratios.

The results are summarized in Figure 3. We observed that $\Delta(f_l(\mathbf{x}_l)) < \Delta(\mathbf{x}_l + f_l(\mathbf{x}_l)) < 1$. Note that we do not say that we should achieve $\mathrm{Var}[\mathbf{x}_0] \to \infty$; our claim is that as long as $\mathrm{Var}[\mathbf{x}_0]$ is nonzero, $\Delta(\mathbf{x}_l + f_l(\mathbf{x}_l))$ obtains a gain closer to one compared to the non-residual block.

Finally, we discard P7, which corresponds to PostDropout (Proposition 2). In summary, based on Propositions 1–4 and the above analyses, we conclude with the following guideline.

Guideline 1. *For each residual block of PreResNet, apply one dropout after the last BN but before the last weight layer, e.g., at P5 or P6.*

2.4 DROPOUT IN HEAD

So far, we have discussed the use of dropout in residual blocks. Additionally, we consider applying dropout in the *head*, which takes the output of the last residual block as the input and outputs the final prediction. Indeed, Bello et al. [2021] observed improved performance when applying dropout after the GAP but before the fully connected layer. This practice has been adopted in several neural networks such as MobileNetV2, EfficientNet, EfficientNetV2, MnasNet, NASNet, and Inception-v4 [Sandler et al., 2018, Tan and Le, 2019, 2021, Tan et al., 2019, Zoph et al., 2018, Szegedy et al., 2017].

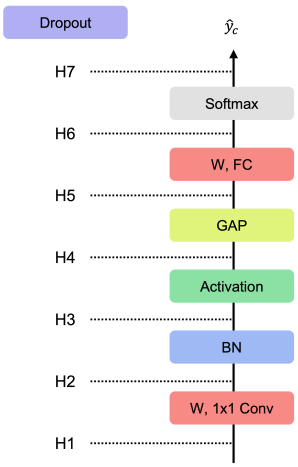

Figure 4: Seven possible positions in head to apply dropout. The composition of the head can vary depending on the model; we illustrate the head commonly deployed in models such as MobileNetV2 and EfficientNet.

In light of this practice, we theoretically investigate the best position in the head to apply dropout. We examine seven possible positions to apply dropout, labeled H1 to H7 (Figure 4). First, owing to the presence of BN in the head, the use of dropout at H1 and H2 should be avoided. For the remaining positions, because there is no subsequent BN, we do not discuss the inconsistency ratio further.

For the head, we now emphasize preventing dropout from resulting in an unstable output. The cross-entropy loss with a one-hot encoded label is $-\log \hat{y}_c$, where $\hat{y}_c$ is the probability on the correct class. For example, applying dropout at H6 or H7 directly drops the predictions, which can result in a lower $\hat{y}_c$ and a significantly larger loss. This large loss can occur even with a correct prediction, resulting in an unstable gradient descent. To obtain a stable loss, it is favorable to have a small variance at the output of the head, thus avoiding H6 and H7. Therefore, we are left with (H3, H4, or H5). Existing practices have preferred to apply dropout at H5; however, we claim that applying dropout at H3 or H4 results in a smaller variance and thus is more advantageous than H5.

Proposition 5. *Applying dropout before GAP (H4) in the head exhibits less variance compared to after GAP (H5):*

$$\mathrm{Var}[\underbrace{\mathrm{GAP}(\mathrm{Dropout}_{\mathrm{train}}(\mathbf{x}))}_{H4}] \tag{34}$$

$$< \mathrm{Var}[\underbrace{\mathrm{Dropout}_{\mathrm{train}}(\mathrm{GAP}(\mathbf{x}))}_{H5}]. \tag{35}$$

The difference arises from the fact that dropout before the GAP masks each element of the feature map, whereas dropout after the GAP masks each channel of the feature map. A detailed proof can be found in the Appendix.

Table 1: Test accuracy on CIFAR dataset. All accuracies in this paper are expressed in percentage units. The difference from baseline performance is presented to the right. * indicates applying dropout following Guideline 1.

	CIFAR-10				CIFAR-100			
	PreResNet-50		PreResNet-110		PreResNet-50		PreResNet-110	
	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference
No Dropout	93.6633	-	94.0300	-	71.3900	-	73.5367	-
P0	84.2500	(-9.4133)	70.9433	(-23.0867)	52.6200	(-18.7700)	18.3467	(-55.1900)
P1	92.9167	(-0.7467)	93.5467	(-0.4833)	70.2000	(-1.1900)	71.2733	(-2.2633)
P2	93.1933	(-0.4700)	93.6867	(-0.3433)	71.1433	(-0.2467)	72.4133	(-1.1233)
P3	93.5833	(-0.0800)	93.8133	(-0.2167)	71.0767	(-0.3133)	72.3967	(-1.1400)
P4	93.2167	(-0.4467)	93.8933	(-0.1367)	70.4800	(-0.9100)	72.1400	(-1.3967)
P5*	**93.8333**	(+0.1700)	**94.4367**	(+0.4067)	72.3633	(+0.9733)	73.6300	(+0.0933)
P6*	93.6767	(+0.0133)	94.2200	(+0.1900)	**72.4267**	(+1.0367)	**73.9800**	(+0.4433)
P7	93.7800	(+0.1167)	94.2667	(+0.2367)	72.0833	(+0.6933)	73.5633	(+0.0267)

Table 2: Experimental results on other conditions.

	Weight Decay 10^{-3}		Weight Decay 10^{-5}		Bottleneck		ELU	
	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference
No Dropout	93.2100	-	92.3733	-	93.7667	-	92.6100	-
Guideline 1	93.3567	(+0.1467)	92.8133	(+0.4400)	93.7933	(+0.0267)	92.7433	(+0.1333)

In summary, we conclude with the following guideline:

Guideline 2. *In the head, apply one dropout after the BN but before the GAP layer, e.g., at H3 or H4.*

3 EXPERIMENTS

3.1 DROPOUT IN RESIDUAL BLOCK

CIFAR Dataset We conducted experiments to observe the performance differences due to the dropout position. First, we compared the performance of PreResNet trained without and with dropout at one of (P0, $\cdots$, P7). We trained PreResNet-$\{50,\ 110\}$ on a multi-class classification task using the CIFAR-$\{10,\ 100\}$ datasets [Krizhevsky, 2009]. See the Appendix for details such as the hyperparameters used. An average of three runs was reported for each result (Table 1).

The experimental results were in agreement with our claims. The greatest accuracy was observed when dropout was applied at P5 or P6, confirming the validity of Guideline 1. Applying dropout at (P5, P6, or P7) resulted in improved accuracy, whereas applying dropout at (P0, P1, P2, P3, or P4) decreased accuracy. This observation implies that the placement of dropout after the second BN matters more than whether it is applied after weight or ReLU. That is, when dropout was applied before the second BN, neither PreDropout nor PostDropout improved the accuracy. An explanation for this phenomenon requires Propositions 3 and 4, and is unique to our analysis compared to the existing literature.

Other Conditions We further validated Guideline 1 using other experimental setups. To test different weight conditions, we varied the weight decay from 10^{-4} to 10^{-3} or 10^{-5}. We also experimented with PreResNet using a bottleneck block, which had three [BN–ReLU–Weight] pipelines, unlike the basic block. In this case, to follow Guideline 1, we applied dropout after the third ReLU but before the third weight layer. We also tested our guideline with ELU [Clevert et al., 2016] to replace ReLU. For the four experimental setups, we observed an improved accuracy for CIFAR-10 and PreResNet-50 (Table 2). Note that applying dropout did not always improve and could actually degrade the performance (Table 1); however, applying dropout in accordance with Guideline 1 consistently and successfully improved performance.

Other Models In addition, we experimented with Guideline 1 using the original ResNetV1, whose residual branch had two [Weight–BN–ReLU] pipelines. In this case, to follow Guideline 1, we applied dropout at the end of the residual branch. We used the ResNetV1-$\{50,\ 110\}$ and CIFAR-$\{10,\ 100\}$ datasets. Again, we observed improved accuracy from dropout using Guideline 1 (Table 3).

Other Datasets We further validated our claim using other datasets. Two datasets were targeted, Caltech-101 and

Table 3: Experimental results using the ResNetV1.

	CIFAR-10				CIFAR-100			
	ResNetV1-50		ResNetV1-110		ResNetV1-50		ResNetV1-110	
	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference
No Dropout	93.2700	-	93.7067	-	70.6200	-	71.8833	-
Guideline 1	93.4600	(+0.1900)	93.9500	(+0.2433)	71.4867	(+0.8667)	73.1133	(+1.2300)

Table 4: Test accuracy on Caltech-101 and Oxford-IIIT Pet datasets.

	Caltech-101				Oxford-IIIT Pet			
	PreResNet-50		PreResNet-101		PreResNet-50		PreResNet-101	
	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference	Accuracy	Difference
No Dropout	83.3212	-	83.8074	-	83.8147	-	84.5668	-
Guideline 1	83.7831	(+0.4620)	84.1964	(+0.3890)	83.9049	(+0.0903)	85.5897	(+1.0229)

Table 5: Experimental results on dropout at the head. "P" represents PreResNet. * indicates applying dropout following Guideline 2.

Dataset	**Model**	**No Dropout**	**H4***	**H5**
Cal	P-50	83.321	83.978	83.005
	P-101	83.807	84.999	83.662
Pet	P-50	83.815	84.988	84.416
	P-101	84.567	85.229	85.259

Table 6: Top-1 accuracy on ImageNet.

Model	**No Dropout**	**H4***	**H5**
MobileNetV2 (1.4)	75.714	75.820	75.718
EfficientNet-B0	77.156	77.240	76.976
ResNet-50	78.834	78.932	78.546
DenseNet-169	79.066	79.152	79.036

Oxford-IIIT Pet [Fei-Fei et al., 2007, Parkhi et al., 2012]. We used PreResNet-$\{50,\ 101\}$ with bottleneck block. See the Appendix for details such as the hyperparameters used. On the two datasets and two PreResNets, we observed that applying dropout following Guideline 1 improved the test accuracy (Table 4).

3.2 DROPOUT IN HEAD

Caltech-101 and Oxford-IIIT Pet We experimented with dropout in the head. We compared three cases: training without dropout and with dropout at one of (H4, H5). We trained PreResNet-$\{50,\ 101\}$ on the Caltech-101 and Oxford-IIIT Pet datasets. The average of three runs was reported for each result (Table 5). We observed that applying dropout at H4 demonstrated greater accuracy than H5. Applying dropout at H5, which is the current consensus in existing studies, improved the accuracy on Oxford-IIIT Pet, yet decreased accuracy on Caltech-101.

ImageNet We further validated our claim using another dataset and models. We targeted ImageNet, a widely used large-scale dataset. See the Appendix for details such as the hyperparameters used. Because our analysis on the head is applicable to any model that employs GAP, we targeted other models: MobileNetV2, EfficientNet, ResNet, and DenseNet. We observed that applying dropout at H4 consistently improved Top-1 accuracy on the ImageNet dataset (Table 6). Note that MobileNetV2 and EfficientNet originally employed dropout at H5; however, we found that it could only marginally influence or even decrease accuracy.

4 CONCLUSION

In this study, we investigated the correct position for applying dropout. We demonstrated that the dropout position influences the variance inconsistency and sought the best position that provides an inconsistency ratio close to one. By analyzing the theoretical properties of the residual networks, we discovered the correct position to apply dropout was after the last BN but before the last weight layer. In several experiments, we observed increased and decreased accuracy depending on the position of dropout, explaining the reason for the performance change using our analysis. In addition, we provided a guideline on applying dropout at the head and validated the improved performance through experiments. We hope that these findings will help practitioners understand and benefit from dropouts.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd (IO201210-08019-01).

References

Irwan Bello, William Fedus, Xianzhi Du, Ekin Dogus Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting ResNets: Improved Training and Scaling Strategies. In *NeurIPS*, 2021.

Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In *ICLR*, 2021.

Shaofeng Cai, Jinyang Gao, Meihui Zhang, Wei Wang, Gang Chen, and Beng Chin Ooi. Effective and Efficient Dropout for Deep Convolutional Neural Networks. *CoRR*, abs/1904.03392, 2019.

Rafaela Castro, Yania Molina Souto, Eduardo S. Ogasawara, Fábio Porto, and Eduardo Bezerra. STConvS2S: Spatiotemporal Convolutional Sequence to Sequence Network for weather forecasting. *Neurocomputing*, 2021.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *ICLR*, 2016.

Soham De and Samuel L. Smith. Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks. In *NeurIPS*, 2020.

Li Fei-Fei, Robert Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 2007.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. DropBlock: A regularization method for convolutional networks. In *NeurIPS*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. In *ECCV*, 2016b.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *CVPR*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift. In *CVPR*, 2019.

Isaak Lim, Anne Gehre, and Leif Kobbelt. Identifying Style of 3D Shapes using Deep Metric Learning. *Comput. Graph. Forum*, 2016.

Brian Liu, Xianchao Xu, and Yu Zhang. Offline Handwritten Chinese Text Recognition with Convolutional Neural Networks. *CoRR*, abs/2006.15619, 2020.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3D Human Pose Estimation in Video With Temporal Convolutions and Semi-Supervised Training. In *CVPR*, 2019.

Hieu Pham and Quoc V. Le. AutoDropout: Learning Dropout Patterns to Regularize Deep Networks. In *AAAI*, 2021.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NIPS*, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a Model for Few-Shot Learning. In *ICLR*, 2017.

Eduardo Romera, Jose M. Alvarez, Luis Miguel Bergasa, and Roberto Arroyo. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Trans. Intell. Transp. Syst.*, 2018.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, 2018.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI*, 2017.

Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, 2019.

Mingxing Tan and Quoc V. Le. EfficientNetV2: Smaller Models and Faster Training. In *ICML*, 2021.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *CVPR*, 2019.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *BMVC*, 2016.

Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online Deep Clustering for Unsupervised Representation Learning. In *CVPR*, 2020.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR*, 2018.